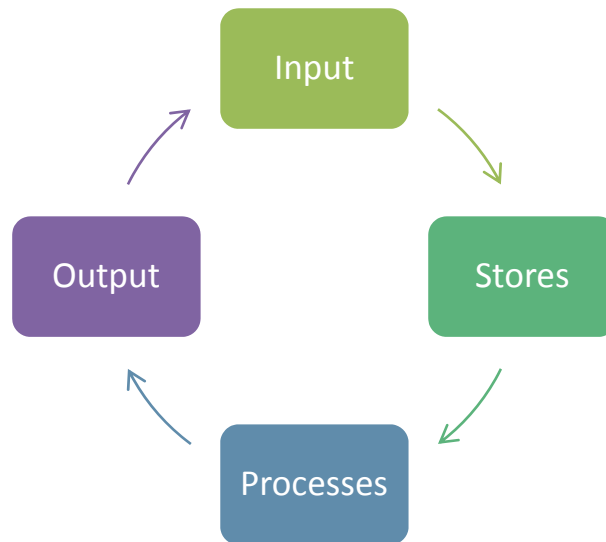


## Programming Language

A computer program does 4 main functions:

1. Takes **INPUT** e.g. from a users keyboard
2. And **STORES** the input in a VARIABLE – *think of it like a storage box*
3. **PROCESSES** or does something with the INPUT so it can
4. Then display the **OUTPUT** on your screen



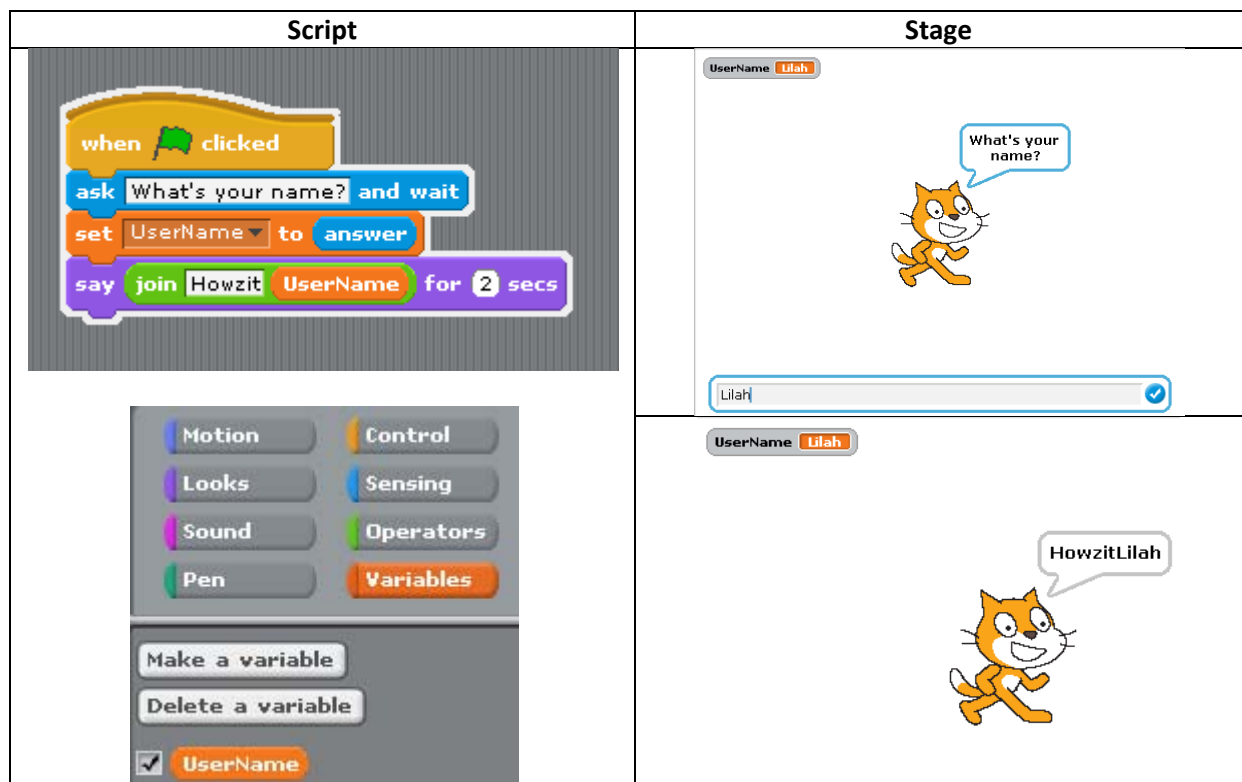
### Tutorial 1 – VARIABLES:

- You will use these in scratch to store input from users
- For example, if I ask you your name. I could store what you typed on the keyboard into a variable with the name - *UserName*.

Lets do it...**Creating a variable and Assigning a value**

1. Open Scratch
2. On the scripts tab, click Control. Drag “When green flag clicked into script”
3. Click Variables \ Make a variable \ Call it *UserName* and click OK – *This created the storage box.*
4. Click on Sensing. Drag the block “ask what’s your name? and wait”
5. Click Variables. Drag the block “set UserNameto” and where it says 0, drag from Sensing \ “answer” – *This is how you assign a value e.g. answer that the user types in answer to the question to the variable you created UserName.*
6. Click Looks \ “say Hello! For 2 secs”.
7. Drag into the first “say hello” position the Operators \ “join hello world” block – *this allows you to join two phrases.*
8. In the first join use a greeting e.g. “Howzit”
9. In the second join position drag variable *UserName* from Variables. *This will greet the user by name for 2 seconds. If this is too quick, increase the time.*

10. Note in the screenshots below, the variable UserName is displayed on the stage. If you don't want this, untick the variable.



Examples of values that can be assigned to a variable *e.g. Score will be 5 until its 5+1, then it will become 6:*

- `Score = 5`
- `Score = Score + 1`
- `PlayerName = "Bob"`
- `Area = length * width`

#### Things to remember with variables:

- Name it something meaningful *i.e. what it is*
- Make it short
- Case matters *e.g. Age is different to age*
- Don't use special characters or spaces

#### Assessment Task 1 – User Greeting:

\*~# 1~31 °æ ~#
















\*3~ ~# 1~3# 6# °æ ~#



Test here

## Tutorial 2 – EXPRESSIONS:

1. Similar to maths, an expression in programming is made up of variables, operators and a value e.g.  $a=b+2$  *a and b are variables; = and + operators and 2 a value.*
2. In Scratch, the operators can be found under Operators –

Name	Description	Scratch Operator(s)	Scratch Example
Addition	Adds values.		
Subtraction	Subtracts a value from another value.		
Increment	Increases a value.		
Decrement	Decreases a value.		
Multiplication	Multiplies values.		
Division (true)	Divides values, giving a result which includes decimals.		
Division (integer)	Divides values, giving an integer as a result – ie anything after the decimal point is not included.	Combination of two blocks:  	



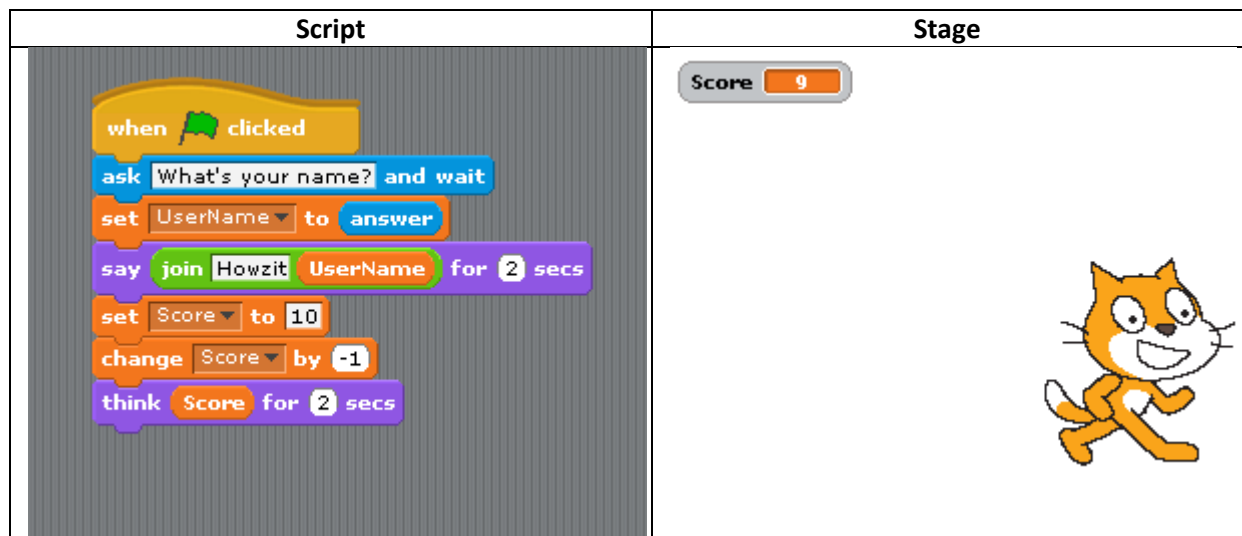
3. The + and – can be used to increase or decrease values assigned to a variable. e.g. *if your variable was SCORE, you would want it to increase until they scored 5 in order to move to the next level, but if they got 2 incorrect answers then you'd want it to reset to 0 and start over.*

## Lets do it...Creating a Variable and Assigning an Expression

1. Create a variable *Score* and make sure its visible and the *UserName* not visible.



2. Set the score to 10 – Variables \ “Set...” block
3. Change the score by -1 – Variables \ “Change...” block
4. Say or Think Score – Looks \ “Think...” or “Say...” blocks
5. What happened to the score? *It should’ve displayed 9 if you had done it correctly*



### Tutorial 3 – PREDEFINED ACTIONS:

1. Scratch offers some predefined actions that save us time. Think of them like a built in shortcut to achieve actions that can be repetitive.
- 2.

### Tutorial 4 – COMMENTS:

1. You will be expected to include comments in your assessment scripts to explain what the code does. This is good practice in case you have to troubleshoot, or someone else looks at your work. It's easier to understand.
2. Right-click anywhere near (but not on) the code blocks. Select add comment. A yellow comment will appear.
3. Type your comment.



## Tutorial 5 – TESTING AND DEBUGGING:

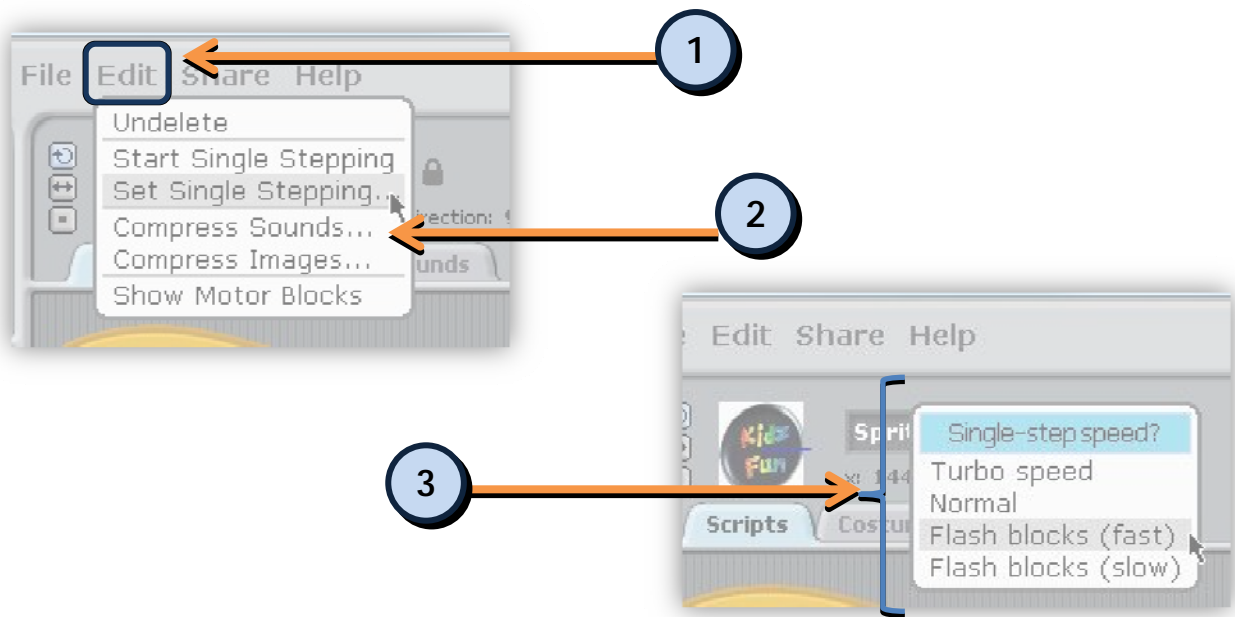
1. This is essential to not only check for errors, but to make sure it runs as intended and performs in the most efficient way possible.
2. Like any language such as English, all programming languages have syntax you must learn if you are going to master the programming language.
3. Because of the blocks that you drag and drop in Scratch it is not possible to generate a syntax error.
4. However there is danger of a Logic Error e.g. the program works, but not as you intend it.
5. For your assessment, you will be expected to show annotated screenshots in a Log that includes a range of tests:
  - a. Valid Input – e.g. *your program accepts valid input. If a number is expected not text.*
  - b. Invalid Input – e.g. *your program recognises invalid input e.g. text was entered when it was expecting a number.*
  - c. Out of range input – e.g. *if user was asked for a postcode of 4 digits, but the user entered 309878 or 2*
  - d. Boundary input – e.g. *the user was asked for a number between 5 and 12. These become the boundary inputs.*
  - e. Exceptional Input – e.g. *the input is valid, but the program behaves unexpectedly.*
6. When you create your Test Plan, consider:
  - a. What are you testing?
  - b. What input has been tested?
  - c. What was the Expected vs Actual result(s)?
7. There is a debugging tool available in Scratch if you ever do run into an error that needs troubleshooting{

## Debugging tools



### ***Single stepping in Scratch***

When you start single stepping, each block performs individually. You can also set it to run the script slower or faster, and to highlight (in yellow) active blocks.



Choose the speed you want for single-stepping.

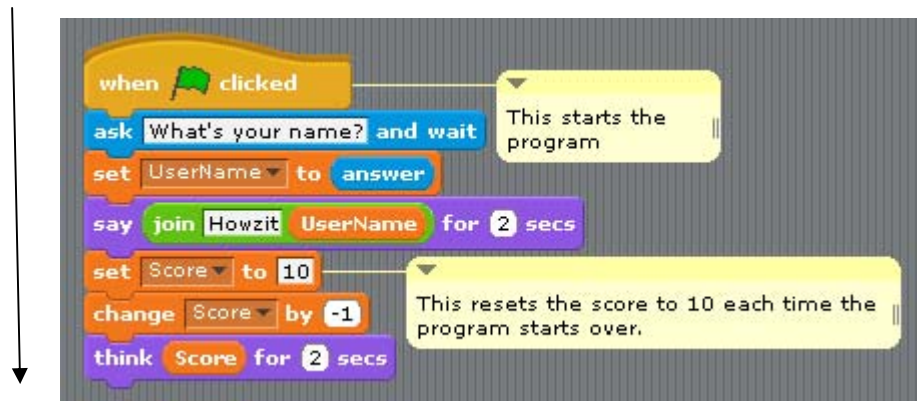
- **Turbo** = very fast
- **Normal** = normal speed
- **Flash blocks (fast)** and **Flash blocks (slow)** = active block is highlighted yellow (see screenshot on the right).



## Tutorial 6 – CONTROL STRUCTURES:

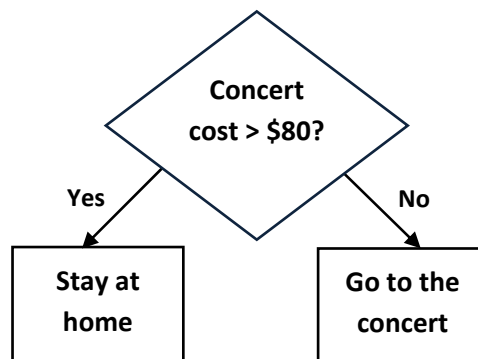
There are 3 main control structures you will be assessed on – Sequence *e.g. steps in order*; Selection *e.g. conditions*; Iteration *e.g. loops*;

- **Sequence:**
  - These are the most basic form of control structure. Your program you've created during this tutorial is an example. The script just runs in order of the blocks from top to bottom.



- **Selection:**

- These will only run IF a condition is true. Selection structures include:
  - i. If...Then
  - ii. If...Then...Else
  - iii. If...Then...Else If
- For example:



- Here, the **condition** 'Does the concert cost more than \$80?' controls which sequence (**action**) is taken. If the **condition** is **true** (ie 'Yes' the concert does cost more than \$80), then the **action** to be taken is 'Stay at home'. If the **condition** is **false** (ie 'No' it does not cost more than \$80), then the **action** to be taken is 'Go to the concert'.
- **Comparison operators** that you can use in Scratch are:

Name	Description	Scratch Operator(s)
Equality	Checks if two things are equal.	
Less than	Checks whether the first value is less than the second value.	
Greater than	Checks whether the first value is greater than the second value.	
Less than or equal to	Checks whether the first value is less than or equal to the second value.	
Greater than or equal to	Checks whether the first value is greater than or equal to the second value.	
Not equal	Checks whether two things are NOT equal.	

## Lets do it...Create a Selection Structure:

Without looking at the script, test your understanding by having a go at creating your own program that follows the algorithm logic below:

### Algorithm Logic

1. Ask your user their name
2. Say hi back to them and say "This is a True or False Game"
3. Tell them to press T or F
4. Ask the question – In summer is eearth closer to the sun than in winter?
5. If the UserAnswer variable equals F – say "Yes that's right"
6. Else – say "Wrong sorry. The earth's distance does not determine seasons."

### Script

```
ask What's your name? and wait
say join Hi join answer ! This is a game of true or false. for 2 secs
say Press T for true or F for false. for 3 secs
ask In summer the Earth is closer to the Sun than in winter. True or False? and wait
set UserAnswer to answer
if UserAnswer = F
  say Yes, that's right for 2 secs
else
  say Sorry, that's wrong. The Earth's distance from the Sun does not determine the seasons. for 5 secs
```

## Lets do it...Create a Gaga vs Swift Game following an Algorithm:

In programming, an algorithm is like a blue print for a house. Never try and design a program without first writing an algorithm that steps you through what the program should do. For your assessment, you will be given a basic algorithm, but if you want merit or excellence you will expand on it yourself. Show Miss your working program and save as Selection.

Say "What do you think of Lady Gaga's latest video?"

Ask "Enter: 1 for awesome; 2 = good; 3 = not great; 4 = hate it!"

Get Answer

IF Answer = 1 THEN

GagaScore = GagaScore + 3

ELSE IF Answer = 2 THEN

GagaScore = GagaScore + 2

ELSE IF Answer = 3 THEN

GagaScore = GagaScore + 1



```

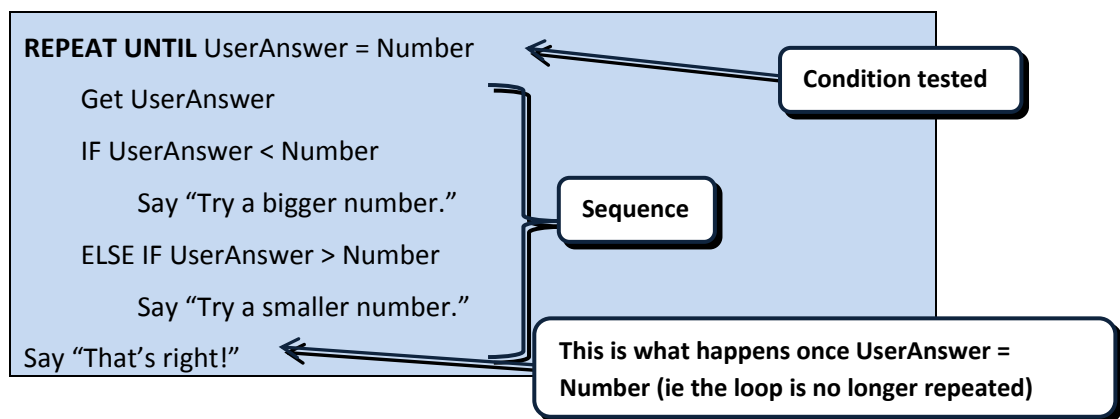
Say "What do you think of Taylor Swift's latest video?"
Ask "Enter: 1 for awesome; 2 = good; 3 = not great; 4 = hate it!"
Get Answer
IF Answer = 1 THEN
    SwiftScore = SwiftScore + 3
ELSE IF Answer = 2 THEN
    SwiftScore = SwiftScore + 2
ELSE IF Answer = 3 THEN
    SwiftScore = SwiftScore + 1
IF GagaScore < SwiftScore
    Say "Taylor Swift wins the head-to-head!"
ELSE IF GagaScore > SwiftScore
    Say "Lady Gaga wins the head-to-head!"
ELSE
    Say "It's a tie!"

```

Lets do it...**Create a Loop:**

- **Iteration:**

- Loops will repeat a sequence a number of times until a condition is met.
- If the condition is tested before a sequence is executed, it's called a **DO WHILE...loop**.
- If the condition is tested after a sequence is executed, it's called a **REPEAT UNTIL...** loop.
- In the example below, the condition *UserAnswer = Number* is tested after the sequence is executed. So is an example of a REPEAT UNTIL ... loop. The loop is repeated UNTIL the condition is met. Try and follow this algorithm and create your own program in Scratch. Show Miss your working program and save it as Loop.



## Lets do it...Login and Password Check:

Using what you've learnt so far in Scratch to do with sequence, selection and iteration structures, try and create the following algorithm in Scratch. Show Miss Edmonds your working program and save it as Password Check:

Set Login to 0

Set Password to 0

REPEAT UNTIL Login = "Bob" or "Kesha"

    Ask "Enter login:"

    Set Login to Answer

    IF Login = "Bob"

        REPEAT UNTIL Password = "!30!3"

            Ask "Enter password:"

            Set Password to Answer

        Say "Welcome BOB!"

    IF Login = "Kesha"

        REPEAT UNTIL Password = "k3\$ha"

            Ask "Enter password:"

            Set Password to Answer

        Say "Welcome Kesha!"

END